IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application No. 09/764,439

Confirmation No. 7079

Applicant: Kaneko et al.

Filed: January 19, 2001

TC/AU: 2174

Examiner: L. Nguyen

Docket No.: 401022

Customer No.: 23548

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Dear Sir:

Applicants request review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a Notice of Appeal.

The review is requested for the reasons stated on the following sheets.

Respectfully submitted,

Phillip M. Pippenger, Reg. No. 46,055
LEYDIG, VOIT & MAYER, LTD.
Two Prudential Plaza
180 North Stetson Ave., Suite 4900
Chicago, Illinois 60601-6731
(312) 616-5600 (telephone)
(312) 616-5700 (facsimile)

Date: August 30, 2007

REASONS FOR BRIEF REQUEST FOR REVIEW

Status of Claims

Claims 6, 11, 12, and 14-23 are pending in this application. Claims 6 and 21 are independent claims. There are no un-entered amendments at this time.

Summary of Claimed Subject Matter

The invention pertains generally to a new system for integrating native and platform independent code in a specific way to facilitate navigation functions. By way of example, claim 6 encompasses a navigation apparatus having four primary elements: (1) hardware and basic functions for controlling the hardware, (2) a ***native*** code navigation application processing block for providing navigation services using these basic functions, (3) an optional ***platform independent*** application processing block for providing optional services, and (4) a ***platform independent*** interface processing block for communicating with the optional application processing block and the navigation application processing block to enable optional services.

Since the optional application processing block calls the interface processing block, the optional application processing block does not need to be compiled based upon each platform. Accordingly, when a modification to the optional application processing block occurs, the time required for compiling is decreased. Consequently, development efficiency is improved. Moreover, since the interface processing block is independent of the optional application processing block, the interface processing block does not need to be compiled upon the compiling of the optional application processing block. In other words, the interface processing block is compiled once because recompilation of the interface processing block is unnecessary, substantially improving development efficiency.

Grounds of Rejection to be Reviewed

The §103 rejection of claims 6, 11, 12, and 14-23 in view of DeLorme, as modified using the IJVM[1] and JNI[2] references.

Reasons for Withdrawal of Rejection

In overview, the cited art, whether taken alone or in combination, plainly fails to teach the specifically recited elements of the claims. For example, claims 6 and 21 both recite a hybrid system wherein a first set of elements of the system is comprised of native code while a second set of elements of the system is comprised of platform independent (or multi-platform) code that interfaces with the elements of the first set. No reference of record shows a hybrid architecture nor indicates the desirability of such an architecture.

Although the actions have identified the existence of both platform-independent and native code types in the art, the existence of these code types in the art does not amount to a suggestion as to the desirability or even the possibility of a specific hybrid system combining both.

In greater detail, claim 6 recites a navigation application[3] that is implemented in native code (i.e., code native to the platform) and an optional application that is implemented in non-native code (i.e., an application that executes on a platform-independent virtual platform). An interface is recited for communicating between the navigation application and the optional application. The interface is itself also implemented in non-native code as recited.

While the Office Action has identified an "interface" in DeLorme, the noted "interface" of DeLorme is not an interface process *between native code and non-native code streams* as recited in the claim. Rather, the cited interface is a data link

[1] "Inside the JAVA Virtual Machine"
[2] "Essential JNI JAVA Native Interface"
[3] Application processing blocks are sometimes referred to here as applications to avoid excessive prose, but this terminology is simply a matter of convenient reference.

between multiple machines. *See* DeLorme at column 8, lines 64-67. ("Alternatively, users can also operate IRMIS 100 from a remote *interface* through wireless or hard-wire links ....") *See also* column 8, lines 41-45. ("This communication *interface* between the portable PDA and home-base desktop ....") In fact, as the Action recognizes, DeLorme fails to teach non-native code entirely (*see* page 3), ***so how could it possibly teach an interface to such code?***

The secondary references do not solve this problem. In particular, JNI and IJVM contain many teachings related to Java, but fail to teach an interface process for ***communicating between running native code and running non-native code providing optional services***. Thus, the cited references, singly and in combination, fail to teach the recited interface.

Moreover, the combination of references is itself not proper. The cited modification of DeLorme in view of IJVM is said to be motivated

> "in order to provide users with a secure, robust, platform independent program(s) to be delivered across networks and run on a great variety of computers and devices."

However, this alleged motivation actually refutes the point for which it is cited. If, as the Action declares, the desire for "robust, platform independent program(s)" would motivate one to exchange native code programs for platform independent code programs, then why would one modify DeLorme *only* in the cited respects while leaving DeLorme's navigation application implemented in native code? Clearly there is a selective and piecemeal modification of references here that has nothing to do with the actual teachings of the references.

Simply put, the references do not support the ***selective*** modification of DeLorme to first replace certain code with platform-independent code, then leave certain code as native code, and then somehow build an unspecified and untaught interface between the two, with the interface itself being platform independent. The references simply do not provide the needed elements, nor do they (or the art generally) provide the requisite motivation to complete the needed modifications.

Conclusion

- The references fail to teach the recited "interface"; the "interface" of DeLorme is not in any way the same as the "interface" defined in the claim. The former is between hardware, while the latter is between specifically defined types of processes.
- The rationale used to combine the references is actually self-defeating. If one would have been motivated to exchange native code programs for platform independent code programs, then why would one modify DeLorme *only* in the cited respects while leaving DeLorme's navigation application implemented in native code?

The Examiner's "Response to Arguments"

- The Examiner questions whether it is permissible to "attack" references individually. However, if *every* reference fails to teach the element in question, then the *combination* also fails to teach that element.

- The Examiner recognizes that DeLorme fails to teach the claimed interface block, but the Examiner asserts that the element is found "inherently" in the reference. However, inherency is a specific legal doctrine, not a mysterious gap filler for plainly missing elements. To be "inherent" in a reference, an element must be ***necessarily*** present given the express teachings of the reference. The Examiner has made no showing that DeLorme inherently includes an interface block between native and platform-independent code. In fact, the Examiner admitted on page 3 of the action that DeLorme "does not explicitly disclose code ...that is platform independent." So how in the world could DeLorme "inherently" teach an ***interface*** to a code type that isn't even ***taught*** by DeLorme? This glaring contradiction begs for an explanation, but the Action provides none.